A MAYA Exporting Plug-in for MPEG-4 FBA Human Characters

Yacine Amara[1], Mario Gutiérrez[2], Frédéric Vexo[2] and Daniel Thalmann[2]

(1) Applied Mathematic Laboratory, EMP (Polytechnic Military School), BP 17 Algiers, Algeria
(2) Virtual Reality Lab (VRlab) EPFL (Swiss Federal Institute of Technology) 1015 Lausanne, Switzerland
Tel: +41-21-693-5216 Fax: +41-21-693-5328
E-mail: Yacine_Amara@hotmail.com, Mario.Gutierrez@epfl.ch, Frederic.Vexo@epfl.ch, Daniel.Thalmann@epfl.ch

Abstract

The aim of this paper is to present a method and a tool to extract the full set of information from an animated character designed in Maya (animatable body only). The Maya plug-in generates the appropriate data as specified in the MPEG-4 FBA standard [2]. The result is composed of three different data sets: the BDP (Body Definition Parameters) describing the character geometry, the BAP (Body Animation Parameters) containing the animation sequence and finally, the BAT (Body Animation Tables) witch contain a deformation mapping.

Keywords

Virtual Characters, MPEG-4 FBA, MAYA plug-in, digital content.

1. Introduction

The MPEG-4 standard [12] was created to unify the coding and description of interactive real-time graphics and distributed multimedia applications. The specification defines uniform ways to describe Cross-media (audio, video, 2D, 3D...etc.) content such as interactive video (digital television), web based (interactive embodied agents) and mobile applications (2D graphics). Our interest focuses on the part concerning the virtual character's definition and representation.

MPEG-4 defines two approaches:

- BBA (Bone Based Animation) for a non-humanoid character definition and animation [3],
- FBA (Face and Body Animation) object describes the geometry of human-like virtual characters and their animation [2] (figure 1).

The MPEG-4 SNHC (Synthetic and Natural Hybrid Coding) group has been working on the MPEG-4 standard for character animation (since 1999 with MPEG-4 version 2), however the standard hasn't been fully accepted by the digital content industries.

One of the problems found with the FBA specification was the limitation to human-like characters and the problems linked to the conversion of rigid models to deformable ones. The geometric description of an FBA character requires the calculation of mesh deformations for each animation frame in order to keep a seamless mesh.

The Bone Based Animation (BBA) framework was proposed as an extension to the FBA to overcome the limitation of characters skeleton (allowing the definition of non-human-like characters), and eliminating the need to pre-calculate the anatomical deformations (skeleton-muscles-based deformation algorithm) [4].

In spite of the continuous improvements, the MPEG-4 character animation still remains in the academic, experimental field. The absence of tools to convert the "in-house" designed characters to the MPEG-4 standard is one of the main obstacles commonly referred by the industry as the main reason for avoiding the adoption of MPEG-4. We decided to tackle this problem by means of implementing a tool capable to produce MPEG-4 compliant virtual characters.

The FBA version was chosen over the BBA due to the flexibility offered by the first one. In the case of BBA, the calculation of mesh deformations is done in an automatic way, based on weights assigned to the vertices close to the character joints, but they aren't flexible enough to fulfill the requirements in terms of expressiveness and/or realism in the animation.

The best is to let the character designer make the fine adjustments to create the exact visual effect he requires to convey emotion and believability. FBA let this freedom to the designer by means of the Body Animation Tables, which store the information about the deformations, independently of the technique used to create them.

FBA has additional advantages: it is less expensive in terms of computation, this qualifies it as the best option for the wider range of applications and the calculations needed for mesh deformations are simple linear interpolations (see details on section 2-3). The choice of characters is limited to humanoids, but the specification remains simple enough to be implemented in a variety of terminals (players) and gives full freedom to the designer.

This work aims at fulfilling the need of the digital content industry concerning the production of synthetic characters. The production companies require reusing and adapting their content to different platforms and applications (the same character can be seen in digital films, mobile applications and interactive web pages). Thus, there is a strong interest to develop a tool to export content created in proprietary software to a standard format widely accepted by the industry.

Digital content producers use a variety of tools for character design and animation. The most popular authoring tools nowadays are Discreet's 3ds Max [9] and Maya from Alias|Wavefront [10]. From these two competitors, we observe that Maya is gaining acceptance in the digital content industry: cinema, games, etc. For example, in the latest computer graphics generated films (Spider man, Ice age, The lord of the rings…) the production team has developed specific animation plug-ins for complex visual effects. Maya is a powerful authoring tool offering as an added value a flexible API and well documented developer's kit.

We have developed a plug-in to export animated characters designed in Maya to the standard MPEG-4 FBA specification, allowing the use of this data in a large application domain.

The rest of this paper is organized as follows: first we describe the basics of the MPEG-4 FBA specification, then we overview the Maya development environment and architecture, after we present the exporting plug-in implementation and finally, we discuss the validation of our results and future work.

2. Description of the MPEG4 FBA

The MPEG-4 standard published by ISO in January 2000, aims to provide an integrated set of tools for compressing and streaming multimedia objects. MPEG-4 standardizes a number of such media objects; capable of representing both natural and synthetic content types [7]. In our work we focus on the representation of human like characters, and exactly about MPEG-4 FBA.

The FBA has a wide area of applications, such as e-commerce, games, virtual teleconferencing and virtual kiosks. MPEG-4 specifies a rich set of FBA parameters to define the model, texture and animation of the face and body. There are two sets of parameters (figure 1) [6]:

- The first set specifies definition of the FBA model: FDPs (Face Definition Parameters) and BDPs (Body Definition Parameters). These parameters allow the decoder to create an FBA model with specified shape and texture.
- The second set defines the animation of the face and body: FAPs (Face Animation Parameters) and BAP (Body Animation Parameters). Typically, FDPs and BDPs are transmitted only once, while the FAPs and BAPs are streamed and coded for each animation frame.

MPEG-4 FBA	
Face Animation	**Body Animation**
Face Definition Parameters **(FDP)**	Body Definition Parameters **(BDP)**
Face Animation Table (FAT)	*Body Animation Table (BAT)*
Face Animation Parameters **(FAP)**	Body Animation Parameters **(BAP)**

Figure 1: FBA parameters.

Our task consists in the development of the MPEG-4 exporting plug-in for human Body. Another partner does the exportation of the animated face. In the rest of this paper we refer only to the body, excluding the animated face.

In the following sub sections we describe in detail each FBA data structure:

2.1 Body definition Parameters

Based on the H-Anim specification [11], the BDP defines a hierarchic structure to describe the geometry of character's body. An H-Anim body contains a set of Joint nodes that are arranged to form a hierarchy (figure 2-1). Each Joint node can contain other Joint nodes, and may also contain a Segment node, which describes the body part associated with that joint. The human body consists of a number of segments (such as the forearm, hand and foot), which are connected to each other by joints (such as the elbow, wrist and ankle). A mesh of polygons will typically define each segment of the body.

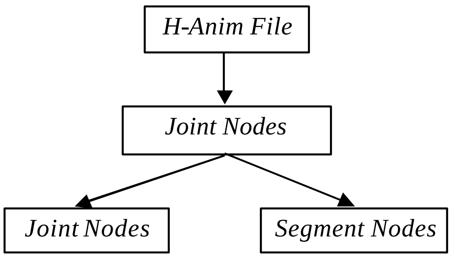

Figure 2-1: Contents of a H-Anim structure.

2.2 Body Animation Parameters

A BAP stream can contain up to 296 parameters describing the topology of the body skeleton [5]. It has three different angle parameters: yaw, roll and pitch. These parameters can be applied to any MPEG-4 compliant body and produce the same animation. If we look at a skeleton as a hierarchy of joints related each other, the 3 possible degrees of rotation for a joint have each a BAP assigned to. The BAP unit is defined as:

BAP = angleInRadians * 100000/p.

2.3 Body Animation Table

Since the BDP model is made out of separated segments, defined by a mesh of polygons, when the rotation angles of the joints are modified by the applied BAPs, the children segments are rotated from its original position. The result is a discontinuous surface over the rotated joint (low visual quality); the set of vertices that form the polygon mesh of the segment need to be modified to keep an anatomically realistic, continuous surface over the modified joint (figure 2-2) [4]. The BAT contains the list of vertices that are modified on each segment, the associated displacement and the list of BAPs, modifying it.

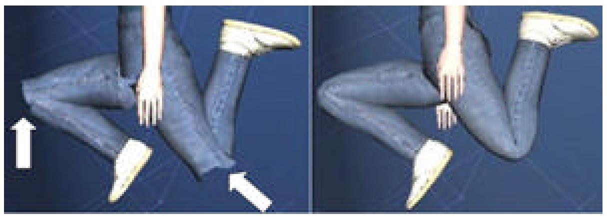

Figure 2-2: Effect of mesh deformations.

3. The MAYA Development Environment

In this section we expose the architecture of Maya. This architecture is organized in four layers: the DG (Dependency Graph) witch regrouped the data structure of the Maya, an API C++ libraries, a script language called MEL (Maya Embedded Language) and the GUI (Graphical User Interface) which is built on top of it (figure 3) [1].

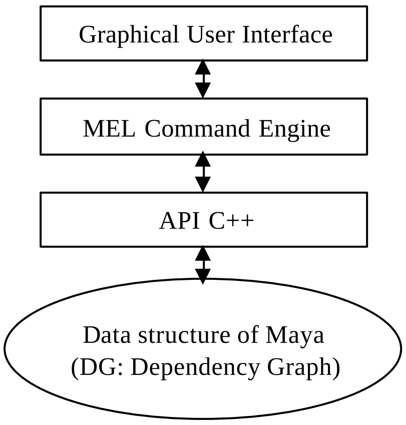

Figure 3 : MAYA system.

Maya has a flexible development tool kit, meaning any user can change existing features or add entirely new ones. There are two ways to add new functionalities to Maya:

- MEL: (Maya Embedded Language) is a powerful and easy to learn scripting language. Most common operations can be done using MEL.
- API: (Application Programmer Interface) provides better performance than MEL. You can add new objects to Maya using API, and code executes approximately ten times faster than when you perform the same task using MEL.

To develop our plug-in we have used Maya API. This choice is motivated by the fact that the MEL doesn't give access to lower level data structure needed to implement the exporting plug-in. additionally, program implemented in C++ executes faster than a MEL script.

The Maya API is packaged as a set of libraries corresponding to different functional areas of Maya. These libraries are:

OpenMaya: Contains fundamental classes for defining nodes and commands and for assembling them into a plug-in.

OpenMayaUI: Contains classes necessary for creating new user interface elements such as manipulators, contexts, and locators.

OpenMayaAnim: Contains classes for animation, including deformers and inverse kinematics.

OpenMayaFX: Contains classes for dynamics.

OpenMayaRender: Contains classes for performing rendering functions.

To build the exporting plug-in, we have used the first three of them. These libraries make it possible to traverse the DG in order to recover the hierarchy of the DAG (Directed Acyclic Graph) containing the nodes, meshes, materials and textures of a character (lower level data structure) [1].

4. Plug-in development

The plug-in is able to export a virtual humanoid designed in Maya to an MPEG-4 standard format. It contains one class with different methods to explore the DG and reorganise the recovered information to build the BDP, BAP and BAT data structure. The input data are a 3D model and an associated animation sequence. In order to create a reusable BAT, the designer must create a generic animation containing the maximum of representative key postures. The class of the exporting plug-in can be divided in three main modules (figure 4):

- BDP structure generation,
- BAP structure generation,
- BAT structure generation.

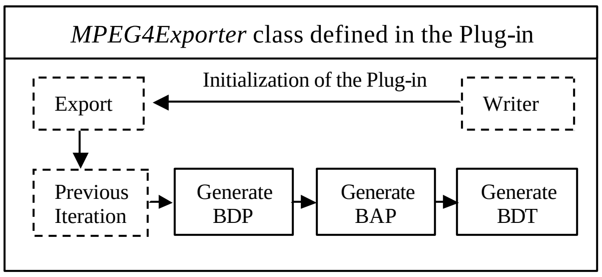
MPEG4Exporter class defined in the Plug-in
Initialization of the Plug-in
Export
Writer
Previous Iteration
Generate BDP
Generate BAP
Generate BDT

Figure 4: Methods and modules diagram of the exporting Plug-in.

4.1 BDP structure generation

In this module, the exporting plug-in gathers all the necessary information to write the BDP File. The virtual human designed in Maya should be formed by one mesh with smooth skin cluster and one skeleton; also it could have one or more materials and textures.

The BDP file will be the H-Anim representation (VRML description) of this virtual human. The virtual human has to be described as a hierarchy of joints that can either have or not an associated segment; moreover, this segments are described as a polygonal set with information about textures and materials. Therefore, to create the file is necessary to have access to the DAG (DG sub graph) in order to get details about the joints, mesh polygons, materials and textures [1]. Moreover, this module is responsible about dividing the virtual human into appropriate segments (figure 4-1).

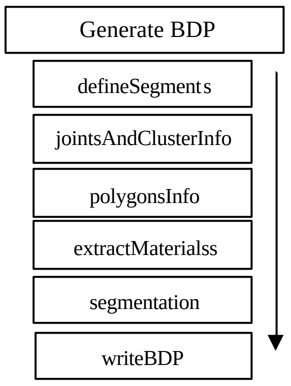

Figure 4-1: Main methods developed in BDP module.

The main methods of this module are:

- **JointAndClusterInfo**: allows segmenting the mesh of the virtual human character into a set of sub-meshes. Each sub-mesh is attached to a joint and is represented by the list of vertices constituting it. The idea is to associate each vertex to one or two joints based on the percentage of participation (skinning weight) of this vertex into these joints [1].
- **PolygonsInfo**: This method gathers information about the polygons that are part of the mesh. It goes through the DAG looking for a mesh and then iterates over its polygons. For each polygon the information that will be stored is: polygon index, list of vertex indexes, list of vertex points, list of UVs, and the material number that colors it.
- **ExtractMaterials**: It is used to look for the materials that each polygon has. It looks in the DG for material characteristics and also for the texture files. Stores the diffuse color, the specular color and the shininess index.
- **Segmentation**: This method uses and links the information gathered through all the previous methods to form the segments that will be written in the BDP file. It allows the verification of collected data coherency by eliminating the redundant data, which is obtained by previous procedures (list of joints with associated vertices, list of polygons, list of materials associated with every polygon). These data will be organized into an appropriate data structure making it easier to write the BDP following H-Anim specifications. This is done in the following procedure:
- **WriteBDP**: This method writes the BDP file using all the data structures that were stored previously. To write the header and the footer is an easy process but to write the body needs a more complex algorithm because the BDP file writing has to follow the H-Anim Hierarchy and the correct VRML structure.

4.2 BAP structure generation

In this module, the exporter obtains all the necessary information to write the BAP File. The animation created in Maya is possible upon changing the rotation values of the selected joints. So, in this module each frame will be examined in order to look for the changes in joint rotation. With this information will be possible to calculate the BAP's and then write them in the BAP File (figure 4-2). The main procedures of this module are:

- **searchJointChanged**: This method will review the joints in the present frame in order to locate and store the angles of the joints rotated.
- **storeBapInfo**: This method is used to store the animation information of every frame in specified structure.
- **writeBapFrame**: This procedure is used to write the animation information of every frame in the BAP file.
- **InitialiseBat**: here we initialize the BAT data structure. This way we get all the information needed to create the BAT, such as the joint name and it's associated segment…etc.

- **getKeyFrames**: This procedure is developed to recover all key frames for any joint. These key frames are defined by the designer to create the animation of the human character.
- **getBatInfo**: this method use the list of key frames associated with each joint. For each call of the method *searchJointChanged*, the procedure attempts to recover information (Baps list, bapCombinations, displacements…etc.) to construct the BAT.

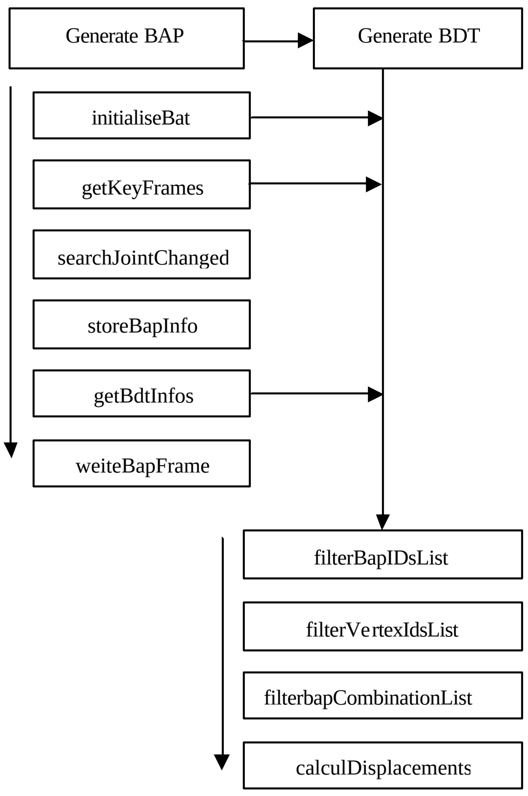

Figure 4-2: Main methods developed in BAP and BAT module.

4.3 BAT structure generation

In this module, the exporter reorganizes and filters the information calculated in *getBatInfos* to write the BAT according to the specification [8] (figure 4-2).

When animating a character, Maya calculates the mesh deformations on the vertices close to the joint's area to keep a seamless surface. The deformations calculated by default can be customized by the designer to achieve realistic results.

The main idea in the BAT is to have a set of pre-calculated deformations corresponding to some key postures for each segment. For instance, the BAT contains only the displacements to apply on the vertices involved in the deformation of the segment (the vertices' initial position correspond to the original, no deformed mesh of the default posture). This set of vertices and their corresponding displacements for each segment and key posture is the core of the BAT. The Maya plug-in calculates the displacements of each segment's vertices at each key posture. This is done through the comparison between the segment's vertices in the initial posture and the vertices deformed by Maya.

5. Validation and Results

The exporting plug-in has been debugged and validated using a web-based MPEG-4 body player [4]. This tool is able to display MPEG-4 compliant files (BDP+BAT+BAP).

Concerning the geometric description (BDP), the player was able to render the character with no difference compared to the original model in Maya. The unique restriction on the geometry is the limit to one mesh for the whole character in Maya. Additional accessories or a segmented mesh will produce unexpected results with the current version of the plug-in; otherwise, all the materials and textures used were correctly exported. The plug-in can export any key-frame animation sequence. The key-frames set by the animator are important when it comes to the generation of the BAT. The last part of this section gives more details on this point which has a strong impact on the visual appearance.

Figure 5-1 shows the Maya models we tested. The "boy" is a 10,000 polygons model and the "girl" is a 20,000 polygons character.

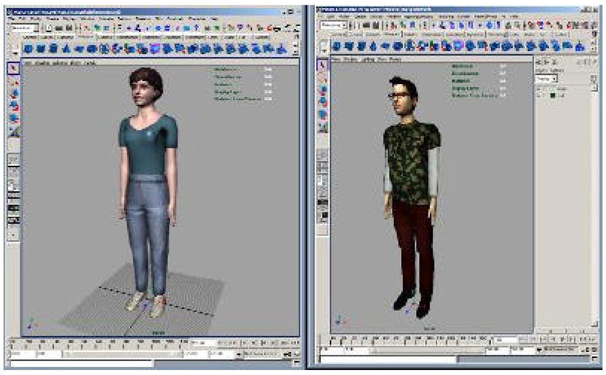

Figure 5-1: The Maya models to be exported.

Figure 5-2 presents two snapshots of the exported model being animated in the web-based application.

As expected, the BAT produced good visual results (accurate deformations) when applied to the original animation –the one used to generate the animation tables. However, the same BAT gave poor results –discontinuities on the mesh- for animations involving key-postures that weren't found in the animation tables.

In order to be reusable in a wide variety of animations, a BAT must contain the whole range of key postures, featuring the extreme positions of the joints as well as intermediate postures for each character's segment. The resolution in the table –the number of key postures- is an important factor to obtain high quality deformations. The best quality possible is obtained when there is a pre-calculated deformation for every character's segment on each frame of animation. However, this would require high amounts of data; the application would be rendering a whole new mesh per frame. For some applications, the quality in anatomical deformations is not a crucial aspect, in these cases; a few representative key-postures can produce acceptable results, as shown in figure 5-2.

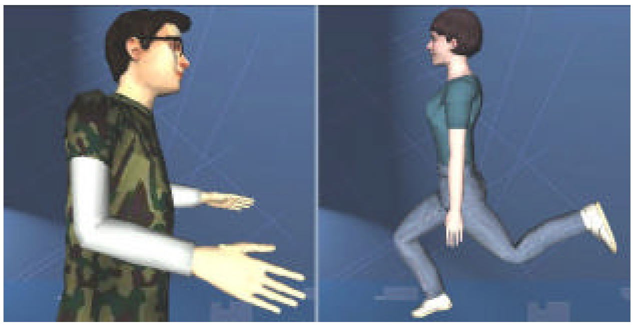

Figure 5-2: The models viewed in a web-based MPEG-4 Body Player.

The absence of pre-calculated deformations for some extreme key postures produces the effect shown in figure 5-3. In both models there is an incorrect or an absence of deformation on the shoulders. These postures weren't present in the animation sequence used to generate the BAT.

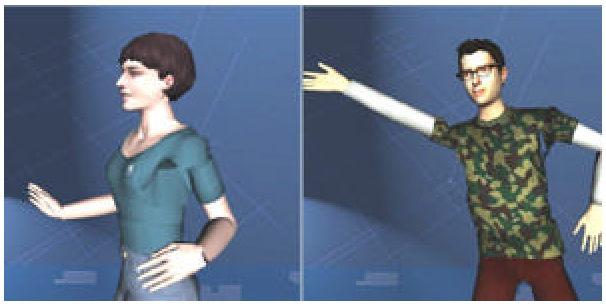

Figure 5-3: Low quality visual results due to absence of adequate key-postures on the BAT.

The selection and generation of adequate and sufficient key-postures is highly dependent of the animation to be displayed. Automating the creation of a generic set of key-postures for a given character would be an added value to the exporting plug-in. However, it is important to notice that the whole process must remain under the designer's supervision, since as mentioned above, high quality and expressive animations still require human intervention –the designer must be able to adjust each key posture on a per vertex basis to assure the best results.

6. Conclusion and future work

We have presented the implementation of a method to produce MPEG-4 FBA animated characters designed in Maya. The resulting data have been tested and validated to be compliant with the specification. The current version of the plug-in accepts only one-mesh-characters; future improvements will include adding procedures to manipulate accessories like glasses, clothes … etc. Integration of the face exporting plug-in to be able to export a full character is also let as future work.

Acknowledgments

This work has been supported by the Swiss Federal Office for Education and Science in the framework of the European project IST-WorldStudio. The license for using Maya was obtained through the Alias/Wavefront Research Donation Program The authors would like to thank Margarita Miranda, Stephanie Noverraz, Thierry Michellod and Sofiane Sarni for their invaluable help.

References

1. *David A. D. Gould.* Complete Maya Programming, An extensive Guide to MEL and the C++ API. *Mogan Kaufmann publishes 2003.*

2. *ISO/IEC JTC1/SC29/WG11, ISO/IEC 14496:1999,* Coding of audio, picture, multimedia and hypermedia information, *N3056, Maui, December 1999*.

3. *M.B. Sévenier et al.* PDAM of ISO /IEC 14496-1 / AMD4, ISO/IEC JTC1/SC29/WG11, *N4415 Dec. 2001, Pattaya.*

4. *M. Gutiérrez, F. Vexo, D. Thalmann,* A MPEG-4 Virtual Human Animation Engine for Interactive Web Based Applications. *Proceedings of the 11th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN 2002), September 2002, Berlin, Germany.*

5. *M. Preda and F. Preteux*, Advanced animation framework for virtual character within the MPEG-4 standard. *Proceedings IEEE International Conference on Image Processing (ICIP'2002), Rochester, NY, 22-25 September 2002.*

6. *T. K. Capin ,Daniel Thalmann.* Controlling and Efficient Coding of MPEG-4 Compliant Avatars. *Proc. IWSNHC3DI'99,Santorini,Greece 1999.*

7. *T. K. Capin, E. Petajan and J. Ostermann.* Efficient modeling of virtual humans in MPEG-4. *IEEE International Conference On Multimedia And Expo (ICME), New York, NY, Volume: 2, 2000.*

8. *T. K. Capin, J. Ostermann and Daniel Thalmann.* Interpolation Function for Deforming Virtual Human surface models based on skeleton joints. *EPFL 2000.*

9. *Discreet, Autodesk, Inc.* 3DS Max,
http://www.discreet.com/products/3dsmax/

10. *Alias|Wavefront.* MAYA, *http://www.aliaswavefront.com/en/products/maya/*

11. *The Humanoid Animation Specification 2001.*
http://www.h-anim.org/Specifications/H-Anim2001/ Web 3D *Consortium Incorporated.*

12. *Overview of the MPEG-4 Standard. http://mpeg.telecomitalialab.com/standards/mpeg-4/ ISO/IEC JTC1/SC29/WG11 N4668March 2002.*